Auton Agent Multi-Agent Syst (2006) 13:429–461
DOI 10.1007/s10458-006-0015-x

Learning collaboration strategies for committees of learning agents

Enric Plaza · Santiago Ontañón

Published online: 9 June 2006
Springer Science+Business Media, LLC 2006

Abstract A main issue in cooperation in multi-agent systems is how an agent decides in which situations is better to cooperate with other agents, and with which agents does the agent cooperate. Specifically in this paper we focus on multi-agent systems composed of learning agents, where the goal of the agents is to achieve a high accuracy on predicting the correct solution of the problems they encounter. For that purpose, when encountering a new problem each agent has to decide whether to solve it individually or to ask other agents for collaboration. We will see that learning agents can collaborate forming *committees* in order to improve performance. Moreover, in this paper we will present a proactive learning approach that will allow the agents to learn when to convene a committee and with which agents to invite to join the committee. Our experiments show that learning results in smaller committees while maintaining (and sometimes improving) the problem solving accuracy than forming committees composed of all agents.

Keywords Multi-agent learning · Committees · Meta learning · Case based reasoning

1 Introduction

A main issue in cooperation in multi-agent systems is how an agent autonomously decides in which situations is better to cooperate with other agents, and with which

E. Plaza · S. Ontañón (✉)
IIIA – Artificial Intelligence Research Institute, CSIC – Spanish Counsel for Scientific Research, Campus UAB, 08193, Bellaterra, Catalonia, Spain
e-mail: santi@iiia.csic.es

Present Address:
S. Ontañón
MAiA – Department of Applied Mathematics, University of Barcelona, Gran Via 585, 08007, Barcelona, Catalonia, Spain

E. Plaza
e-mail: enric@iiia.csic.es

agents does the agent cooperate. Specifically in this paper we focus on multi-agent systems composed of learning agents (each one in principle with a different background), where the goal of the agents is to achieve a high accuracy on predicting the correct solution of the problems they encounter. For that purpose, when encountering a new problem each agent has to decide whether to solve it individually or to ask other agents for collaboration. When taking those decisions, the agent has to consider whether, for each specific problem, collaborating with other agents is likely to improve the prediction accuracy.

This generic scenario can be exemplified in the domain of marine biology where the difficult task of identifying marine sponges prompted us to address it as a multi-agent system. The basic issue to be addressed is that no biologist expert in limnology and benthology has complete knowledge of all forms and kinds of marine sponges. In practice, biologists expert in marine sponges collect specimens on their own on different parts of the world, developing a partial expertise depending on location and species. Moreover, some species change according to the location they live. Therefore, there will be no single biologist that is an expert in all kinds of sponges and all kind of oceans: they refer this fact as people having different backgrounds. When one of the biologists finds a new sponge, he needs to identify specific species of the new sponge. However, since she is not an expert on all kinds of sponges, often she will realize she has low confidence in determining the correct species of the new sponge. In this context, the expert will likely ask other biologists for counsel on the correct species of the new sponge. Notice that the biologist has to take two decisions: (a) deciding when she better asks counsel to other biologists (i.e. when to collaborate with other biologists), and (b) decide which of the other biologists he will ask counsel to.

In this paper we propose a multi-agent system approach to deal with such scenarios. In our approach, we will consider that each biologist has a learning agent that has access to all the marine sponges collected (and properly classified) by her. Each learning agent is able to identify a new sponge based on previous experience. However, if the agent cannot produce a prediction with high confidence, it can decide to collaborate with other agents. Thus, (analogously to the biologist behavior) the agent has to take two decisions: when to start collaboration with other agents, and with which of the other agents to collaborate. Taking these two decisions properly is crucial, since the correctness of the predictions made by the learning agent strongly depends on them.

One of our goals is to show that, through collaboration, individual learning agents and multi-agent systems can improve their performance. Both learning and collaboration are ways in which an agent can improve individual performance. In fact, there is a clear parallelism between learning a collaboration in multi-agent systems, since they are ways in which an agent can deal with its shortcomings. Table 1 shows the main motivations that an agent may have to learn or to collaborate.

Therefore, learning and collaboration are very related. In fact, with the exception of motivation to collaborate in the fourth row of Table 1, they are two extremes of a continuum of strategies to improve performance. An agent may choose to increase performance by learning, by collaborating, or by finding an intermediate point that combines learning and collaboration in order to improve performance. Specifically, we are interested in studying how an individual learning agent can improve its performance by collaborating with other agents, and how can a

Table 1 Motivations of an agent to learn and/or co–operate

Motivations to learn	Motivations to collaborate
Increase quality of solutions	Increase quality of solutions
Increase efficiency	Increase efficiency
Increase the range of solvable problems	Increase the range of solvable problems
	Have access to resources that only other agents can use

learning agent decide whether it is better to work individually or to cooperate with others.

Moreover, returning to the motivating example, we model the situation when a group of biologists collaborate to identify a given marine sponge as the institution we usually call *committee*. When a committee of human biologists is formed, the individual biologists have their own background and produce their individual predictions. We consider a committee goes through two main phases: discussion and deliberation. During discussion, different alternatives are presented and arguments justifying or attacking alternatives are exchanged; during deliberation, one of the alternatives is chosen by some voting system that determines the winner. Analogously, we use the notion of electronic institutions [11] for a group of learning agents collaborating to reach a join prediction, i.e. forming a *committee of agents*. In this paper we will focus only on the deliberation phase, although current work on the argumentative phase is published elsewhere [24].

Specifically, there are two core aspects we want to address in this paper, namely when a committee is needed or not, and which agents should be invited to join a committee. Notice that, in our biology scenario, a biologist often solves the sponge identification task individually, while, some other times recourse to external counsel (in our model, convenes a committee) because she estimates she is not competent with respect to the problem at hand. Our approach to this issue is equipping the agents with a *competence self-model* capable of estimating if the agent is competent (or to which degree it estimates might be competent) to solve a specific problem. Moreover, we will show how this *competence self-model* can be individually learnt by every agent in the course of its regular process of solving problems and collaborating with other agents.

Concerning the second issue, notice that when a biologist decides to consult some other biologists, she does not call all the available sponge experts in the world, but just a small sample, enough to correctly identify the new sponge. For a multi-agent system this means that an agent convening a committee will not simply invite always all agents in the system to join the committee. Thus, we distinguish two types of strategies for convening committees: *fixed committees* and *dynamic committees*. An agent convenes fixed committees when the agents invited to join the committee are always the same regardless of the problem to be solved (an example of this strategy is the basic one of always convening *all* the available agents to a committee). Moreover, when an agent convenes a dynamic committee it has to select which agents to invite in function of the problem to be solved. In this paper we propose to equip each individual agent with *competence models* of the other agents; these competence models assess the confidence of the convener agent in that some other agent is competent to solve the problem at hand. Moreover, we will show how these *competence models* can be individually learnt by every agent in the course of its regular process of solving problems and collaborating with other agents (Sect. 3.1).

1.1 Committees and machine learning

Committees allow us to study the application of machine learning techniques to multi-agent systems, and the relation between collaboration and learning. From a machine learning perspective, a committee may be considered an ensemble of agents, where each agent plays the role of a predictor (trying to predict the correct solution for a given problem). Ensembles of predictors are expected to have a higher performance than individual predictors because of the *ensemble effect* [26]. The ensemble effect is well known in machine learning, and states that, given that some preconditions are met, the combination of predictions made by several individual predictors is likely to be more accurate than the prediction made by the individual predictors. The preconditions of the ensemble effect are simple: each individual predictor must be minimally competent (i.e. have an error rate lower than 0.5) and the ensemble must be diverse (i.e. the error correlation between the predictions of the individual classifiers must be low).

In previous work [27], we have shown that committees of agents can also benefit from the ensemble effect, as ensembles of predictors do. Thus, by properly defining strategies to convene committees, agents can convene committees that allow them to achieve higher performance than working individually. However, committees are not the same as ensembles, in other words our goal is not to present new ensemble learning methods. The fundamental differences between committees and ensembles are, for instance, that autonomy and privacy are not an issue in ensemble learning, but they are essential in multi-agent systems. Moreover, in an ensemble, the ensemble learning algorithm is *centralized* and *creates* the individual predictors in such a way that the ensemble works; however, in a committee, agents are not created by a centralized process, agents are in principle created or maintained by different organizations in different places; therefore, in our multi-agent framework an agent has to convene a committee that achieves the maximum performance by collaborating with the existing agents, and having no control or access to the data they have stored locally. These hypotheses of *decentralized control* and *distributed data* that our framework espouses are not satisfied by ensemble learning methods that assume centralized control and access to data. Therefore ensemble learning methods are not directly applicable to committees, although committees can use the core ideas of the "ensemble effect" to improve their performance [27].

1.2 An approach to learning to cooperate

The problem of convening dynamic committees is presented in this paper inside a framework called *Multi-agent Case Based Reasoning Systems* ($\mathcal{M}$AC) [27]. A $\mathcal{M}$AC system is composed of a set of CBR agents, where a CBR agent is an agent that uses Case Based Reasoning (CBR) [1] to solve problems and learn from those problems. The open and dynamic nature of multi-agent systems fits with open and dynamic nature of *lazy learning* [2] used in CBR. This framework is quite general and has been used to study different aspects concerning learning in multi-agent systems [19–23, 27, 28].

In this paper, however, we focus on presenting *collaboration strategies* that the agents in a $\mathcal{M}$AC system can use to convene committees in the phase of join deliberation (and therefore excluding the argumentation phase). For this reason, we will use a $\mathcal{M}$AC system where agents learn to perform a classification tasks without lose of generality. In machine learning, a classification task is one where a learning system predicts one item among a set alternatives (usually called classes). Since we are focusing on the deliberation phase of committees, where one of the presented alternatives has to be selected, this task is, from the point of view of the learning agent, a classification task. Notice our approach is general in the sense that the alternatives under deliberation can be internally complex (e.g. a committee can deliberate on alternative plans of action), but in the deliberation phase the individual agents have just to predict (and learn to predict) the better alternative.

Specifically, we will present a basic strategy called the Committee Collaboration Strategy (CCS) that always convenes a committee using all the available agents in the system. CCS is a strategy for *fixed committees* and is used for comparison purposes since using all agents in a committee will (in principle) lead to more accurate predictions. After that, we will present another strategy called Proactive Bounded Counsel Collaboration Strategy (PB-CCS), that tries to achieve accurate predictions, but only convening committees when required. Thus, PB-CCS is our proposal to address the problem of deciding when to collaborate, and with which agents to collaborate. We will present specific *decision policies* that agents may use to decide when to solve problems individually and when to convene committees, and to select which agents to invite to join a committee. Moreover, the key claim of this work is that agents can learn to make those decisions (when to collaborate and with which agents to collaborate). To support this claim, we present a proactive learning approach that gives the agents the ability to learn how to take those decisions.

Since those decision policies are based on what we call *competence models*, we will present two approaches: one where the competence models are predetermined (and manually build by the agent designers) and another one where those competence models are individually learnt by every agent in the systems using PB-CCS. We also present experiments to compare the performance of fixed versus dynamic committees, and that of learning competence models versus predetermined competence models.

The structure of the paper is as follows. First Sect. 2 presents the multi-agent framework in which we have performed our experiments, and formally define the notion of committee. Moreover, Sect. 2 presents the CCS. After that, Sect. 3 introduces the notion of *dynamic committees*, and the PB-CCS, that will be presented a dynamic committee collaboration strategy. Then, Sect. 4 presents a proactive learning technique with which agents will be able to learn a decision policy used to convene dynamic committees. Sect. 5 formally presents the B-CCS for comparison purposes. Finally, Sect. 6 presents an empirical evaluations of all the collaboration strategies in several scenarios. The paper closes with related work and conclusions sections.

2 A multi-agent CBR approach

In this paper we focus on agents that use CBR to solve problems. CBR techniques are suitable to multi-agent systems and give the agents the capability of autonomously learn from experience by retaining new cases (problems with known solution). Therefore, we will focus on *Multi-agent CBR Systems* ($\mathcal{M}\mathsf{AC}$).

Definition 2.1 A *Multi-Agent Case-Based Reasoning System* $\mathcal{M} = \{(A_1, C_1), \ldots, (A_n, C_n)\}$ is a multi-agent system composed of a set of agents $\mathcal{A} = \{A_i, \ldots, A_n\}$ where each agent $A_i \in \mathcal{A}$ possesses an individual case base C_i, and each agent A_i uses CBR with its case base C_i but has neither access nor control over other case bases.

A case base $C_i = \{c_1, \ldots, c_m\}$ is a collection of cases, and a *case* $c = \langle P, S\rangle$ is a tuple containing a case description $P \in \mathcal{P}$ and a solution $S \in \mathcal{S}$, where $\mathcal{P}$ is the problem space and $\mathcal{S}$ is the solution space. In this paper, we assume that $\mathcal{S} = \{S_1, \ldots, S_K\}$ is a finite and known set of solutions. Notice that case descriptions are defined over the problem space $\mathcal{P}$. In the following, we will use the terms *problem* and *case description* indistinctly. Therefore, we can say that a case consists of a case description plus a solution, or that a case is a problem/solution pair. In the following, we will use the dot notation to refer to elements inside a tuple. e.g. to refer to the solution of a case c, we will write $c.S$. Moreover, we will also use the dot notation with sets, i.e. if C is a set of problems, $C.P$ refers to the set of problems contained in the cases in C, i.e. $C.P = \{c.P | c \in C\}$.

In our framework, interaction among agents is realized by means of *collaboration strategies*, that we define using the methodology of electronic institutions [11].

Definition 2.2 A *collaboration strategy* $\langle I, D_1, \ldots, D_m\rangle$ defines the way in which a group of agents in a $\mathcal{M}$AC collaborate in order to achieve a common goal and is composed of two parts: an interaction protocol I, and a set of individual decision policies $\{D_1, \ldots, D_m\}$.

The *interaction protocol* of a collaboration strategy defines a set of interaction states, a set of agent roles, and the set of actions that each agent can perform in each interaction state. The agents use their individual *decision policies* to decide which action to perform, from the set of possible actions, in each interaction state. Each agent is free to use its own decision policies. Moreover, we have used the ISLANDER formalism [10] to specify the interaction protocols in our framework (Fig. 1).

Let us now define the notion of *committees of agents* that allows a group of agents to collaborate in solving problems with the goal that the committee performance improves with respect to that of solving problems individually. A *Committee* is a group of agents that join together to predict the solution of a problem P. Each agent individually predicts the solution of P and then all the individual predictions are aggregated by means of a voting process.

The only requirement on the CBR method that an agent uses is that it must be able to provide a collection containing the most relevant cases for the current problem,

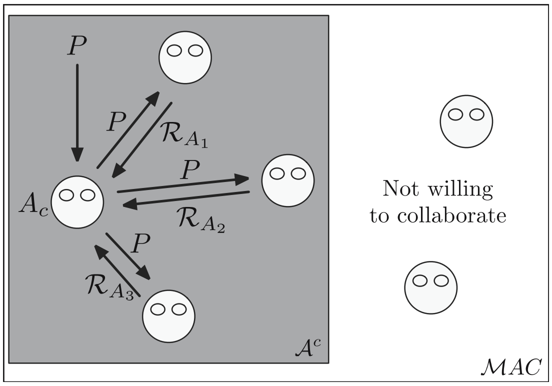
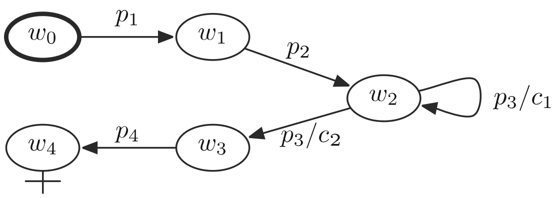


Fig. 1 Illustration of a $\mathcal{M}$AC system where an agent A_c is using CCS in order to convene a committee to solve a problem



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, \mathcal{A}^c, !P)$
p_3/c_1 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert !w_0 w_1 \mathbf{R}\rvert < \#(\mathcal{A}^c) - 2$
p_3/c_2 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert !w_0 w_1 \mathbf{R}\rvert = \#(\mathcal{A}^c) - 2$
p_4 :	$Inform(!A_i, !User, ?S)$

Fig. 2 Interaction protocol for the CCS

that we will call the *retrieval set*. From this retrieval set, agents construct Solution Endorsement Records (SERs). A SER is a tuple $\mathbf{R} = \langle S, E, P, A\rangle$ where A is an agent that has found E (where $E > 0$ is an integer) cases endorsing the solution S as the correct solution for the problem P in the retrieval set. Intuitively, a SER is a record that stores the result of the individual retrieval of cases. If the retrieved cases belong to more than one solution then a different SER will be built for each solution.

Let us present now the CCS.

Definition 2.3 The *Committee Collaboration Strategy* (CCS) is a collaboration strategy $\langle I_C, D_V\rangle$, where I_C is the CCS interaction protocol shown in Fig. 2 and D_V is a decision policy based on the BWAV voting system, presented in Sect. 2.1.

The interaction protocol I_C is described in Fig. 2 using the ISLANDER [10] formalism and applies to a set of agents $\mathcal{A}^c$ that have agreed to join a committee. The protocol consists of five states and w_0 is the initial state. When a user requests an agent A_i to solve a problem P the protocol moves to state w_1. Then, A_i broadcasts the problem P to all the other agents in the system and the protocol moves to state w_2. Then, A_i waits for the SERs coming from the rest of agents while building its own SERs; each agent sends its SERs to A_i in the message p_3. When the SERs from the last agent are received the protocol moves to w_3. In w_3, A_i will apply the voting system defined in the individual decision policy D_V (with all the SERs received from other agents and the SERs built by itself) to aggregate a joint prediction. Finally, the aggregate prediction S will be sent to the user in message p_4 and the protocol will move to the final state w_4.

Notice that not all the agents in the $\mathcal{M}$AC system may be willing to collaborate using CCS. Therefore, the set $\mathcal{A}^c$ contains only those agents that are willing to collaborate, as shown in Fig. 1.

Since all the agents in a $\mathcal{M}$AC system are autonomous CBR agents, they will not have the same problem solving experience. Therefore, the content of their case bases will not be (in principle) the same. For that reason, not all the agents will be able to correctly solve the same problems. In other words, the individual agents' errors are (in principle) not correlated. Thus, using CCS, prediction accuracy can be improved because the convened committee of agents satisfy the preconditions of the ensemble effect.

2.1 Bounded weighted approval voting

Agents could in principle use any voting system to aggregate their predictions. However, in this paper we use a voting system called bounded weighted approval voting (BWAV) specifically designed for committees of CBR agents. As we will see in Sect. 4, agents will perform a learning process that will use as input the information provided by the votes of the agents. Thus, the more informative are the votes, the better the agents will be able to learn. For that reason, BWAV is more adequate than standard *majority voting* (where each agent will simple vote for a single solution).

The principle behind BWAV is that agents vote for a solution depending on the number of cases in the retrieval set that endorse that solution. Specifically, each agent has one vote that can be assigned to a unique solution or fractionally assigned to a number of classes depending on the number of endorsing cases for each solution.

Let $\mathcal{R}^c = \{\mathbf{R}_1, \ldots, \mathbf{R}_m\}$ be the set of SERs built by the n agents in $\mathcal{A}^c$ to solve a problem P. Notice that each agent may submit one or more SERs. In fact, an agent will submit as many SERs as different solutions are present in the retrieval set. Let $\mathcal{R}_{A_i} = \{\mathbf{R} \in \mathcal{R}^c | \mathbf{R}.A = A_i\}$ be the subset of SERs of $\mathcal{R}$ created by the agent A_i. The vote of agent A_i for a solutionS_k is the following:

$$Vote(S_k, P, A_i) = \begin{cases} \frac{\mathbf{R}.E}{c+N} & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

where c is a normalization constant that in our experiments is set to 1 and $N = \sum_{\mathbf{R} \in \mathcal{R}_{A_i}} \mathbf{R}.E$ is the size of the retrieval set of A_i. Notice that if an agent A_i has not created a SER for a solution S_k, then the vote of A_i for S_k will be 0. However, if A_i has created a SER for S_k, then the vote is proportional to the number of cases found endorsing the solution S_k.

We can aggregate the votes of the agents in $\mathcal{A}^c$ for one solution S_k by computing the ballot: $Ballot(S_k, P, \mathcal{A}^c) = \sum_{A_i \in \mathcal{A}^c} Vote(S_k, P, A_i)$ and therefore, the winning solution is the solution with more votes in total:

$$Sol(\mathcal{S}, P, \mathcal{A}^c) = \arg\max_{S_k \in \mathcal{S}} Ballot(S_k, P, \mathcal{A}^c) \tag{2}$$

BWAV can be seen as a variation of *Approval Voting* [3]. The main difference between approval voting and BWAV is that in *Approval Voting* each agent votes for all the candidates they consider as an acceptable outcome without giving weights to the accepted options.

3 Dynamic committees

The CCS can effectively improve the problem solving performance of the agents in a $\mathcal{M}$AC system with respect to agents solving problems individually [25]. However, when an agent uses CCS, no policy is used to select which agents are invited to join the committee and *all* the agents in a $\mathcal{M}$AC system are invited each time that an agent wants to use CCS. Moreover, it is not obvious that forming a committee with all the available agents is the best option for all the problems: possibly smaller committees have an accuracy comparable (or indistinguishable) to that of the complete committee. Furthermore, possibly some problems could be confidently solved by one agent while others could need a large committee to be solved with confidence.

In this paper, we want to provide a collaboration strategy that is capable of: (a) deciding when an individual agent can solve a problem individually and there is no need to convene a committee, and (b) when a committee is needed, deciding which agents should be invited to join the committee. A collaboration strategy that convenes a different committee in function of the current problem is called a *Dynamic Committee* collaboration strategy. Moreover, as we have stated in Sect. 1, agents require *competence models* in order to decide when to convene a committee and which agents to invite.

3.1 Competence models

Competence models are used by agents to decide if a committee is needed and which agents to invite to join a committee. A *competence model* is a function that estimates the confidence on the prediction of an agent (or set of agents) for a specific problem P, i.e. estimates the likelihood that the prediction is correct.

Competence models can be acquired by two different ways: (a) directly specified by a human designer, (b) automatically learned from experience by the agents. In this paper we will present a proactive learning process to allow agents to learn their own competence models. Moreover, in the experimental results section we will compare the learned competence models against handcrafted competence models.

Competence models will be used in the iterative process of convening dynamic committees. Specifically, will be given three main uses: (a) competence models are used to allow an agent to decide whether to convene a committee or not, (b) given that committees are formed iteratively (i.e. agents are invited one by one), competence models are specifically used to decide whether another agents is needed or not in the current committee, and (c) finally, competence models are also used to decide which is going to be the next agent to be invited to join the committee. Moreover, an agent solving a problem individually will be seen as a committee formed of a single agent.

Assessing the confidence on of a committee on a problem P is tantamount to assess how agents have responded to P, therefore competence models assess the competence of agents or groups of agents given a voting situation, i.e. a situation in which committee has been convened and the agents have provided their individual predictions for P. Notice that the collection of SERs $\mathcal{R}_{\mathcal{A}^c}$ casted by the agent members of a committee $\mathcal{A}^c$ completely characterizes a voting situation (since from $\mathcal{R}_{\mathcal{A}^c}$ we can obtain which agents are members of the committee and which have been their votes). Therefore, we will define a *voting situation* $\mathcal{R}_{\mathcal{A}^c}$ as the set of SERs for a problem P sent by a committee of agents $\mathcal{A}^c$ to the convener agent (including the SERs of the convener agent A_c).

For each voting situation we can define the *candidate solution* $S^c = Sol(\mathcal{S}, P, \mathcal{R}_{\mathcal{A}^c})$ of a voting situation as the solution that the committee will predict if no more agents join the committee. Moreover, we can also define the individual candidate solution of an agent A_i in a committee $S^c_{A_i} = Sol(\mathcal{S}, P, \mathcal{R}_{A_i})$ as the solution that A_i individually predicts for the problem.

The specific competence models required to convene dynamic committees for an agent A_i member of a $\mathcal{M}$AC system composed of n agents $\mathcal{A} = \{A_1, \ldots, A_n\}$ are the following: $\mathcal{M}_{A_i} = \{M_c, M_{A_1}, \ldots, M_{A_{i-1}}, M_{A_{i+1}}, \ldots, M_{A_n}\}$, where M_c is a *Committee-Competence Model* and M_{A_j} are *Agent-Competence Models*.

A *Committee-Competence Model* M_c is a competence model that assesses the confidence in the prediction of a committee $\mathcal{A}^c$ in a given voting situation $\mathcal{R}$. Thus, M_c is used to decide whether the current committee $\mathcal{A}^c$ is competent enough to solve the problem P or not (and therefore it is better to invite more agents to join the committee).

An *Agent-Competence Model* M_{A_j} is a competence model that assesses the confidence in the prediction made by an agent A_j in a given voting situation $\mathcal{R}$. M_{A_j} is useful for the convener agent to select which agent A_j is the best candidate to be invited to join the committee by selecting the agent A_j for which its competence model predicts the highest confidence (i.e. the agent with the highest likelihood that its prediction is correct) given the current voting situation $\mathcal{R}$.

Notice that the convener agent A_i requires a self-competence model to decide whether to solve a problem P individually or convene a committee. For this purpose A_i solves the problem P using its own case base, and constructs the corresponding SERs. A collection of SERs is precisely a voting situation, and therefore we can consider that when A_i solves problems individually he is convening a committee of one agent. For this reason, self confidence can be assessed just as before, using the Committee-Competence Model M_c over the voting situation of this committee of one. Therefore, A_i will also use M_c as the self-competence model.

3.2 Proactive bounded counsel collaboration strategy

The PB-CCS is designed to study if the decisions that have to be taken to convene dynamic committees can be learnt. Specifically, agents using PB-CCS will engage in a proactive process to acquire the information they need in order to learn the competence models that allows them to decide when to convene or not a committee, and which agents to invite to join each committee.

Before explaining the proactive learning process, we will first explain how a dynamic committee is convened. For this purpose, we propose an iterative approach to determine the committee needed to solve a problem. The iterative approach works as follows: In the first round, only the convener agent individually predicts the solution of the problem. Then, a competence model is used to determine whether there is enough confidence on the individually predicted solution. It there is enough confidence, then no committee is needed, and the prediction made by the agent is considered the final solution. This first step implements the decision on whether or not a committee has to be convened for the problem at hand. The *Halting* decision policy will be used to take this decision. However, if there is not enough confidence, then a committee is convened, and a new agent A_j is invited to join the committee.

Once a committee is established, at each round we the *Halting* decision policy to decide whether there is enough confidence on the solution predicted by the current committee or not. In the negative case, we use the he *Agent Selection* decision policy to select a new agent to be invited. Figure 3 illustrates this process: from all the agents in the $\mathcal{M}$AC system that have agreed to collaborate, some of them have already joined the committee, and some of them are candidates to be invited if the confidence in the solution predicted by the current committee is not high enough. Moreover, notice that some agents in the $\mathcal{M}$AC system may be unwilling (for whatever reason) to participate in PB-CCS, thus are not candidates to be invited to join the committee.

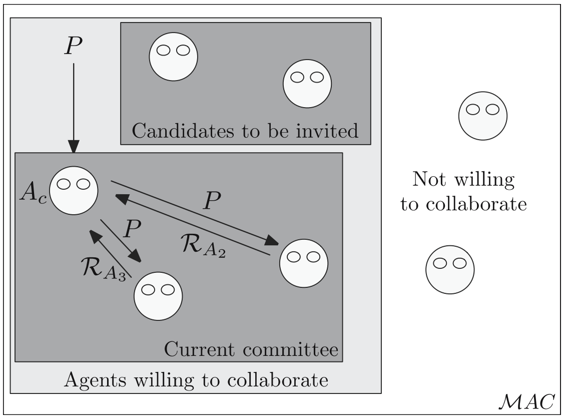
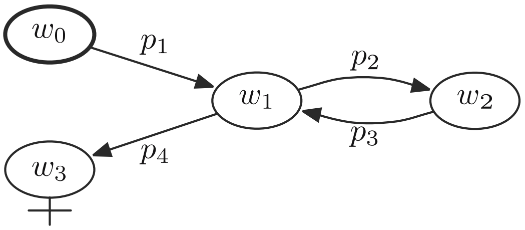


Fig. 3 Illustration of PB-CCS where 3 agents have already been invited to join the committee, forming a committee of 4 agents



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, ?A_j, !P)$
p_3 :	$Inform(!A_j, !A_i, ?\mathbf{R})$
p_4 :	$Inform(!A_i, !User, ?S)$

Fig. 4 Interaction protocol for the *Proactive Bounded Counsel* collaboration strategy

Definition 3.1 The *Proactive Bounded Counsel Collaboration Strategy* (PB-CCS) is defined as a collaboration strategy $\langle I_B, D_H, D_{AS}, D_V \rangle$, consisting of an interaction protocol I_B, shown in Figure 4, D_H is the Proactive Bounded Counsel *Halting* decision policy, D_{AS} is the Proactive Bounded Counsel *Agent Selection* decision policy, and D_V is the voting decision policy based on BWAV (see Sect. 2.1).

PB-CCS is an iterative collaboration strategy consisting in a series of rounds. We will use t to note the current round of the protocol; thus, $\mathcal{A}_t^c$ will be the subset of agents of $\mathcal{A}$ that have joined the committee at round t and $\mathcal{A}_t^r$ the subset of agents of $\mathcal{A}$ that have not yet been invited to join the committee at round t. Finally, we will note $\mathcal{R}_{\mathcal{A}_t^c}$ the set of all the SERs submitted to the convener agent by all the agents in $\mathcal{A}_t^c$ (included the SERs built by the convener agent A_c itself), i.e. $\mathcal{R}_{\mathcal{A}_t^c}$ represents the voting situation at round t.

Figure 4 shows the formal specification of the I_B interaction protocol. The protocol consists of four states: w_0 is the initial state, and, when a user requests an agent A_i to solve a problem P, the protocol moves to state w_1. The first time the protocol is in state w_1 the convener agent uses the D_H decision policy to decide whether to convene a committee or not.

When a committee is convened, the D_{AS} decision policy is used to choose an agent A_j, and message p_2 is sent to A_j containing the problem P. After that, the protocol moves to state w_2. A_i remains in state w_2 until A_j sends back message p_3 containing its own prediction for the problem P, and the protocol moves back to state w_1. In state w_1 the convener agent A_i assesses the confidence of the current voting situation and uses the D_H decision policy to decide whether another agent has to be invited to join the committee or not. If A_i decides to invite more agents, then message p_2 will be send to another agent (chosen using the D_{AS} decision policy), repeating the process of inviting a new agent; if A_i decides that no more agents need to be invited to join the committee the voting system specified in D_V will be used to aggregate a joint prediction S. Finally, A_i will send the joint prediction to the user with message p_4, and the protocol will move to the final state w_3.

3.3 Proactive bounded counsel decision policies

Using the competence models defines in Sect. 3.1, we can define the *Proactive Bounded Counsel Halting* decision policy D_H as a Boolean decision policy that decides whether the convener agent can stop inviting agents to the committee at a round t; i.e. if $D_H(\mathcal{R}_{\mathcal{A}_t^c}) = true$, no more agents will be invited to join the committee. Specifically D_H is defined as follows:

$$D_H(\mathcal{R}_{A_i}) = \Big(M_c(\mathcal{R}_{\mathcal{A}_t^c}) \geq \eta_1\Big) \vee \Big(max_{A_j \in \mathcal{A}_t^r}(M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})) < \eta_2\Big)$$

where η_1 and η_2 are threshold parameters. The rationale of this policy is the following: if the confidence in the solution predicted by the current committee is high enough ($M_c(\mathcal{R}_{\mathcal{A}_t^c}) \geq \eta_1$) there is no need to invite more agents since the current prediction has a very high confidence. Moreover, if the confidence on an agent $A_j \in \mathcal{A}^r$ that is not in the committee is very low ($M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c}) < \eta_2$) inviting A_j to join the committee is not advisable (since the prediction of that agent will very likely be incorrect and would increase the chances that the committee prediction is incorrect). Therefore, if the maximum confidence of every agent in $\mathcal{A}_t^r$ is very low, i.e. $max_{A_j \in \mathcal{A}_t^r}(M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})) < \eta_2$, inviting any of these agents to join the committee is not advisable. This follows from a preconditions of the ensemble effect (see Sect. 1), the one stating that individual members of an ensemble have to be minimally competent.

The two threshold parameters η_1 and η_2 have the following interpretation: η_1 represents the minimum confidence required for the committee's prediction (candidate solution) of the current voting situation; η_2 represents the minimum confidence required in the prediction of an individual agent to allow that agent to join the committee.

Notice that by varying η_1 and η_2, the behavior of PB-CCS can be changed. Assuming that by adding more agents to the committee the confidence of the predicted solution will tend to increase, if we set a high value for η_1, the convener agent will tend to convene larger committees; and if we set a low value for η_1, the convener agent will stop inviting agents earlier, since a lower confidence will be considered adequate enough. Moreover, by setting a high value for η_2, the convener agent will be very selective with the agents allowed to join the committee (since only those agents with a confidence higher than η_2 will be allowed to join). On the other hand, a low value of η_2 will make the convener agent very permissive, and any agent could potentially be invited to join the committee.

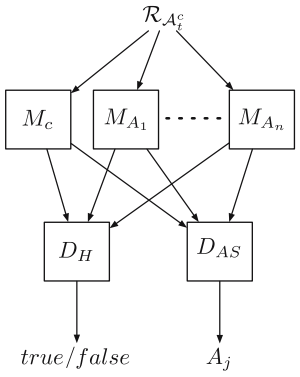


Fig. 5 Relation among the competence models and the Proactive Bounded Counsel decision policies

In fact, if $\eta_1 = 0.0$, an agent will always solve problems individually, and if the parameters are set to $\eta_1 = 1.0$ and $\eta_2 = 0.0$ the resulting collaboration strategy will always convene all the available agents in the $\mathcal{M}$AC system, and therefore achieve the same results than CCS. Furthermore, by increasing η_2 (leaving $\eta_1 = 1.0$) we obtain a collaboration strategy that invites to join the committee all those agents that have a confidence level higher than η_2. Therefore, η_1 and η_2 allow us to define a whole range of different strategies to build committees.

The second decision policy is the *Proactive Bounded Counsel Agent Selection* decision policy $D_{AS}(\mathcal{R}_{A_i}, \mathcal{A}_t^r) = argmax_{A \in \mathcal{A}_t^r}(M_A(\mathcal{R}_{\mathcal{A}_t^c}))$. That is to say, D_{AS} takes as input a voting situation $\mathcal{R}_{A_i}$ and a set of candidate agents to be invited to the committee and determines which is the agent that has the highest confidence on finding the correct solution for a given problem.

Figure 5 shows the relations among the competence models and the decision policies in PB-CCS. The figure shows that at each round, the current voting situation, $\mathcal{R}_{\mathcal{A}_t^c}$, is the input to the competence models. Then, the output of the competence models are used as the inputs of the decision policies.

4 Proactive learning

This section presents a proactive learning technique with which an agent A_i in a $\mathcal{M}$AC system can learn its competence models $\mathcal{M}_{A_i}$ to be used in PB-CCS. In order to learn these competence models, agents need to collect examples from where to learn. This section presents the way in which an agent can proactively collect those examples and how can competence models be learnt from them.

The proactive learning technique consists of several steps (shown in Fig. 6): first, an agent A_i that wants to learn a competence model M obtains a set of cases (that can be taken from its individual case base). Then, these cases are transformed to problems (by removing their solutions) and are sent to other agents in order to obtain their individual predictions for them. After that, with the predictions made by the other agents for all the problems sent, A_i will construct a set of voting situations. Finally, these voting situations will be the input of a learning algorithm from which the competence models will be learnt.

In order to easily apply machine learning techniques, we need to characterize the voting situations by defining a collection of attributes in order to express them as

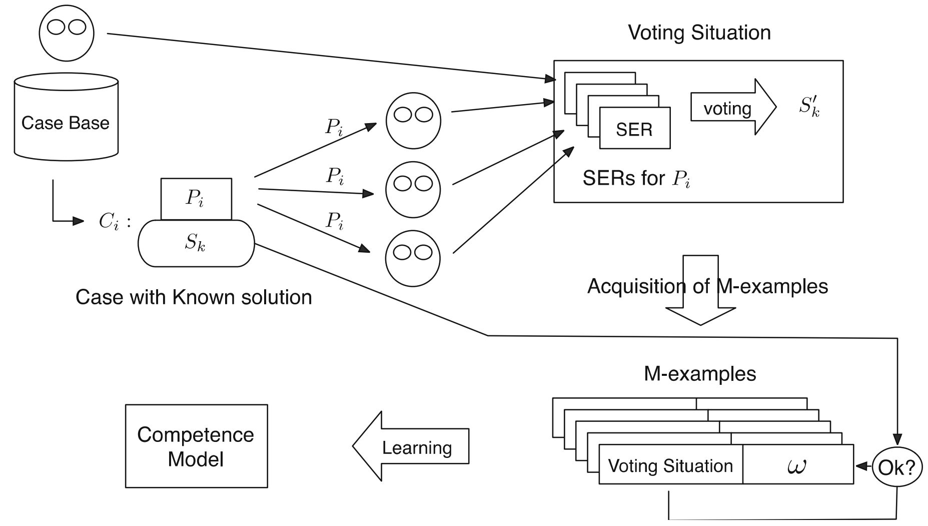


Fig. 6 Detailed graphical representation of the proactive learning technique to learn competence models

attribute-value vectors. The *characterization of a voting situation* $\mathcal{R}_{\mathcal{A}_t^c}$ is a tuple with the following attributes:

– The attributes $A_1, \ldots, A_n$ are Boolean. $A_i = 1$ if $A_i \in \mathcal{A}_t^c$ (i.e. if A_i is a member of the current committee), and $A_i = 0$ otherwise.
– $S^c = Sol(\mathcal{S}, P, \mathcal{R}_{\mathcal{A}_t^c})$ is the candidate solution.
– $V^c = Ballot(S^c, \mathcal{A}_t^c)$ are the votes for the candidate solution.
– $V^r = (\sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}_t^c)) - V^c$ is the sum of votes for all the other solutions.
– $\rho = \frac{V^c}{V^c + V^r}$ is the ratio of votes supporting the candidate solution.

We will use $\upsilon = \langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle$ to note the characterization of a voting situation. Moreover, an *M-example m* is a pair $m = \langle \upsilon, \omega \rangle$, where υ is the characterization of a voting situation $\mathcal{R}_{\mathcal{A}_t^c}$ and ω represents the "prediction correctness" of the voting situation, such that $\omega = 1$ if the candidate solution of the voting situation $\mathcal{R}_{\mathcal{A}_t^c}$ was the correct one and $\omega = 0$ otherwise.

4.1 Acquisition of *M-examples*

Since an agent A_i needs to learn several competence models, a different training set T_M will be needed to learn each competence model $M \in \mathcal{M}_{A_i}$. We will call $\mathcal{T}_{A_i} = \{T_{M_c}, T_{M_{A_1}}, \ldots, T_{M_{A_{i-1}}}, T_{M_{A_{i+1}}}, \ldots, T_{M_{A_n}}\}$ to the collection of training sets needed by an agent A_i to learn the competence models $M \in \mathcal{M}_{A_i}$.

Specifically, an agent A_i that wants to obtain the collection of training sets needed to learn the competence models proceeds as follows:

1. A_i chooses a subset of cases $B_i \subseteq C_i$ from its individual case base.
2. For each case $c \in B_i$:
   (a) A_i uses I_C (the interaction protocol of CCS) to convene a committee of agents $\mathcal{A}^c$ to solve the problem $c.P$. After this, A_i has obtained the SERs built by all the rest of agents in $\mathcal{A}^c$ for problem $c.P$.

(b) A_i solves $c.P$ using a *leave-one-out* method[1] and creates its own set of SERs $\mathcal{R}_{A_i}$.

(c) With the set $\mathcal{R}_{\mathcal{A}^c}$ of SERs obtained in step (a) and (b), A_i builds a number of voting situations from which to construct M-examples (as explained below).

Notice that A_i can build more than one voting situation from the collection $\mathcal{R}_{\mathcal{A}^c}$ of SERs in Step 2.(c). For instance, the set of SERs built by A_i, $\mathcal{R}_{A_i} \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where only agent A_i has cast votes. The set of SERs built by A_i and any other agent A_j, $(\mathcal{R}_{A_i} \cup \mathcal{R}_{A_j}) \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where A_i and A_j have cast their votes. In the following, we will write $\mathcal{R}_{\mathcal{A}'}$ to refer to the set of SERs built by a set of agents $\mathcal{A}'$.

A *Valid Voting Situation* $\mathcal{R}_{\mathcal{A}'}$ for an agent A_i and a problem $c.P$ is a voting situation where A_i has casted its votes, i.e. a set of SERs built by a set of agents $\mathcal{A}'$ that at least contains A_i. Specifically, $\mathcal{R}_{\mathcal{A}'} \subseteq \mathcal{R}_{\mathcal{A}^c}$ such that $\mathcal{A}' \subseteq \mathcal{A}^c$ and $A_i \in \mathcal{A}'$. Intuitively, a valid voting situation for an agent A_i is one in which A_i itself is a member of the committee. Therefore, a valid voting situation can be built by selecting the set of SERs built by any subset of agents $\mathcal{A}' \subseteq \mathcal{A}^c$ (such that $A_i \in \mathcal{A}'$). We can define the set of all the possible subsets of agents of $\mathcal{A}$ that contain at least A_i as $\mathbb{A}(A_i) = \{\mathcal{A}' \in \mathcal{P}(\mathcal{A}) | A_1 \in \mathcal{A}'\}$, where $\mathcal{P}(\mathcal{A})$ represents the parts of the set $\mathcal{A}$ (i.e. the set of all the possible subsets of $\mathcal{A}$). Now it is easy to define the set of all the possible Valid Voting Situations for an agent A_i that can be constructed from $\mathcal{R}_{\mathcal{A}^c}$ as follows:

The *Set of Valid Voting Situations* for an agent A_i is: $\mathbb{V}(A_i) = \{\mathcal{R}_{\mathcal{A}'} | \mathcal{A}' \in \mathbb{A}(A_i)\}$, where $\mathcal{R}_{\mathcal{A}'}$ represents the set of SERs built by the set of agents $\mathcal{A}'$. Using the previous definitions, we can decompose Step 2.(c) above in three sub-steps:

1. A_i takes a sample of all the possible Valid Voting Situations that can be built: $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ (see below).
2. For every voting situation $\mathcal{R} \in \mathbb{V}'$, the agent A_i determines the characterization of the voting situation $\langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle$.
3. With this characterization A_i can build M-examples. Specifically, A_i will build one M-example for each competence model $M \in \mathcal{M}_{A_i}$.

Let us now focus on how M-examples are constructed for each specific competence model $M \in \mathcal{M}_{A_i}$:

- To build an M_c-example, A_i determines the candidate solution $S^c = Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'})$ obtained by applying the voting system to $\mathcal{R}_{\mathcal{A}'}$. If $Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'}) = c.S$, then the following M_c-example is built: $m = \langle\langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle, 1\rangle$ where $\omega = 1$ because the M-example characterizes a voting situation where the predicted solution is correct. If $S^c \neq c.S$, then the following M_c-example is built: $m = \langle\langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle, 0\rangle$ where $\omega = 0$ because the M-example characterizes a voting situation where the prediction is not correct.
- To build an M_{A_j}-example, A_i determines the individual candidate solution yield by A_j, i.e. $S^c_{A_j} = Sol(\mathcal{S}, c.P, \mathcal{R}_{A_j})$. If $S^c_{A_j} = c.S$ (i.e. the prediction of A_j is correct), then the following M_{A_j}-example is built: $m = \langle\langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle, 1\rangle$ and if $S^c_{A_j} \neq c.S$ (i.e. the prediction of A_j is incorrect), then the following M_{A_j}-example is built: $m = \langle\langle A_1, \ldots, A_n, S^c, V^c, V^r, \rho \rangle, 0\rangle$.

[1] The leave-one-out method works as follows: the agent A_i temporally removes the case c from its case base; then it solves tries to solve the problem description $c.P$ using the rest of cases.

Notice that with each voting situation $\mathcal{R} \in \mathbb{V}'$, an M-example can be constructed for each different competence model in $\mathcal{M}_{A_i}$. Therefore, the larger the size of $\mathbb{V}' \subseteq \mathbb{V}(A_i)$, the larger the number of M-examples that can be constructed. The size of $\mathbb{V}(A_i)$ (that is equivalent to the size of $\mathbb{A}(A_i)$) depends on the number of agents in the committee convened to solve each of the problems $c.P$ (where $c \in B_i \subseteq C_i$). In fact, the size of $\mathbb{V}(A_i)$ grows exponentially with the size of the set of convened agents: there are 2^{n-1} different Valid Voting Situations for a $\mathcal{M}$AC system with n agents. Therefore, building all the M-examples (i.e. a "complete data set") that can be derived from all possible valid voting situations in $\mathbb{V}(A_i)$ may be unfeasible or impractical. For that reason, an agent using the proactive learning technique to learn competence models will take a sample $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ of the complete data set.

The number of M-examples that an agent builds for each competence model M is about $\#(B_i) \times \#(\mathbb{V}')$ (where the $\#(A)$ notation represents the cardinality of A). The number of M-examples that agents need to collect for learning appropriate competence models may vary in function of the application domain. In general, the more M-examples collected, the better. However, collecting many M-examples will waste resources of the agent, thus in function of the resources an agent is wiling to spend in building competence models, the number of M-examples to collect must be determined. In our experiments we have imposed the limit of at most 2000 M-examples for each competence model. Therefore, the agents in our experiments will take subsets $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ to have at most $2000/\#(B_i)$ voting situations. Moreover, in our experiments, an agent A_i using the proactive learning technique uses all the case base C_i as the set B_i (i.e. $B_i = C_i$) (in order to maximize the diversity in the set of voting situations built), and therefore the size of $\mathbb{V}'$ will be at most $2000/\#(C_i)$ (see [25] for a more informed method to build the sample $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ than simply taking a random sample).

4.2 Learning of competence models

Once an agent A_i has collected enough *M-examples*, competence models can be learnt. In our experiments we have used an induction algorithm similar to decision trees [29] that we call *confidence trees*.

Since competence models predict confidence values (i.e. real numbers in the interval $[0, 1]$), decision trees cannot be directly used to obtain confidence values (since they predict class labels). For that reason, we are going to define *confidence trees*, a structure consisting on two types of nodes: *decision nodes* and *leaf nodes*. Decision nodes contain conditions, and leaf nodes contain three real numbers: p_l^-, p_l, and p_l^+ (such that $p_l^- \leq p_l \leq p_l^+$); where p_l is the expected confidence in that a voting situation that is classified in a leaf l will yield a correct candidate solution, and p_l^- and p_l^+ are respectively, the pessimistic and optimistic estimations of that confidence.

The technique to learn confidence trees is like that of building decision trees [29], but with the following considerations:

1. Numerical attributes are discretized. Each numeric attribute a is discretized to have just two possible values. The discretization is performed by computing a threshold κ. Left branch of the decision node will have the M-examples with $value(a) \leq \kappa$ and in the right branch all the M-examples which $value(a) > \kappa$.
2. Error-based pruning [5] of the tree is used to avoid overfitting (i.e. for not learning a too specific decision tree that overfits the particularities of the training set).

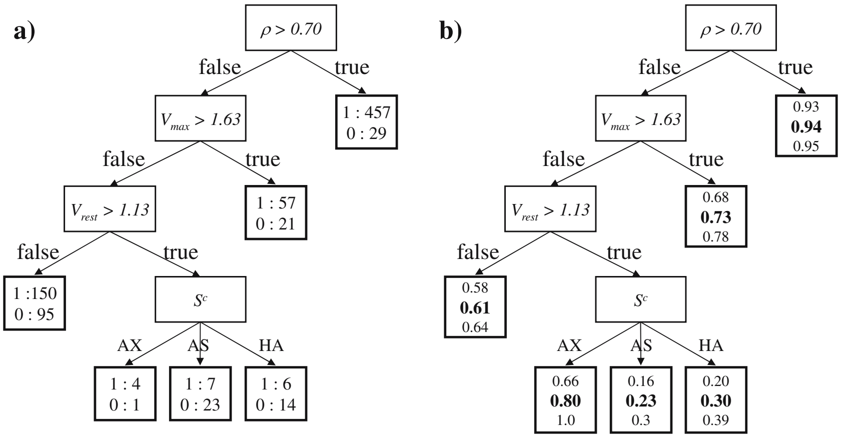


Fig. 7 (**a**) Tree where each leaf contains the number of M-examples with $\omega = 1$ and $\omega = 0$. (**b**) Final confidence tree learnt as the competence model M_C in a $\mathcal{M}$AC system composed of 5 agents. AS, AS and HA are the possible solution classes in $\mathcal{S}$

3. For every leaf node we will store the number of examples of each solution that fall in that leaf. Figure 7(a) shows a tree such that in each leaf l, the number of M-examples with $\omega = 1$ and with $\omega = 0$ is shown. For instance, let us consider the right-most leaf in the tree, a node that contains all examples satisfying $\rho > 0.70$; the figure shows that there are 457 examples with $\rho > 0.70$ and $\omega = 1$, and 29 examples with $\rho > 0.70$ and $\omega = 0$.
4. Finally, the leaf nodes of the confidence tree are computed as follows: let a_l be the number of M-examples with $\omega = 1$ and b_l the number of M-examples with $\omega = 0$ in a given leaf l of the tree. Then, the values corresponding to the leaf l take the following values:
   – $p_l = (1/(a_l + b_l)) * (1 * a_l + 0 * b_l)$ is the expected confidence of an M-example classified in leaf l.
   – p_l^-: the pessimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).
   – p_l^+: the optimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).

Figure 7(b) shows an example of a confidence tree where on each leaf l the three values p_l^-, p_l, and p_l^+ are shown. Since p_l is just an estimation of the confidence, if the number of M-examples in the leaf node l is small then p_l may be a poor estimator of the confidence of the candidate solution of voting situations classified on the leaf l. The greater the number of M-examples in leaf l, the better the estimation of the confidence. To solve this problem, instead of estimating the confidence as a single value, the agents will compute an interval, $[p_l^-, p_l^+]$, that ensures with 66% certainty that the real confidence value is in that interval. This interval depends on the number of examples in leaf l: the greater the number of M-examples, the narrower the interval will be (those intervals can easily computed numerically using basic bayesian probabilistic computations). In Figure 7(b), p_l^- and p_l^+ are shown above and below p_l respectively. For instance, if we look at the right most leaf in Fig. 7 (the one with 457 M-examples with confidence 1 and 29 M-examples with confidence 0), we can see that the estimated p_l is 0.94 and the interval is $[0.93, 0.95]$, a very narrow interval since the number of M-examples to estimate the confidence is high.

For the purposes that competence models will have in the dynamic CCS, pessimistic estimation is safer than any other estimation (expected p_l or optimistic p_l^+). Using pessimistic estimations the worst that can happen is that the committee convened to solve a problem is larger than in could be. However, if we make a more optimistic estimation of the confidence, (using the expected p_l or optimistic p_l^+ estimations) the convener agent may stop inviting agents too early, thus failing to correctly solve a problem more often. Therefore, since confidence trees will be used as competence models, p_l^- will be used as the output of the competence model, i.e. the output of a competence model M for a voting situation $\mathcal{R}_{\mathcal{A}^c}$ is $M(\mathcal{R}_{\mathcal{A}^c}) = p_l^-$, where l is the leaf of the confidence tree in which the voting situation $\mathcal{R}_{\mathcal{A}^c}$ has been classified.

5 Bounded counsel collaboration strategy

In this section we are going to define a non-learning approach to form dynamic committees, the B-CCS. B-CCS works basically in the same way than PB-CCS, but uses predefined competence models instead of learnt ones. Thus B-CCS is only presented with the goal of comparing it with PB-CCS. The B-CCS is composed by an interaction protocol and two decision policies:

Definition 5.1 The *Bounded Counsel Committee Collaboration Strategy* (B-CCS) is a collaboration strategy $\langle I_B, D_H, D_V \rangle$, where I_B is the B-CCS interaction protocol shown in Fig. 4, D_H is the Bounded Counsel *Halting* decision policy (used to decide when to stop inviting agents to join the committee), and D_V is the voting decision policy based on BWAV (see Sect. 2.1).

B-CCS uses I_B, the same protocol as PB-CCS. Moreover, when a new agent is invited to join the committee in B-CCS, a random agent A_j is selected from the set of agents that do not belong to the committee. Thus, B-CCS requires only an individual decision policy: the Bounded Counsel *Halting* decision policy D_H, that decides whether inviting more agents to join the committee is needed.

The D_H decision policy uses the *C-Competence* model that measures the confidence in a solution predicted by a committee to be correct.

$$C\text{-}Competence(\mathcal{R}^c) = \begin{cases} \frac{1}{M} Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c) & \text{If } N > 1, \\ min(Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c), 1) & \text{If } N = 1. \end{cases}$$

where M = $\sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}^c)$, is the sum of all the votes casted by the agents and N = $\#(\{S_k \in \mathcal{S} | Ballot(S_k, \mathcal{A}^c) \neq 0\})$, is the number of different classes for which the agents have voted for.

That is to say, if the agents in $\mathcal{A}^c$ have built SERs for a single solution ($N = 1$), the *Committee-Competence* model will return the ballot for that solution. Moreover, notice that the ballot for a solution when there are more than one agent in $\mathcal{A}^c$ can be greater than 1. Therefore we take the minimum between the ballot and 1 to ensure that the competence models output confidence values within the interval $[0, 1]$. The intuition is that the higher the ballot, the larger the number of cases retrieved by the agents endorsing the predicted solution, and therefore the higher the confidence on having predicted the correct solution. Moreover, if the agents in $\mathcal{A}^c$ have built SERs for more than one solution (and therefore $N > 1$), the *C-Competence* model will return the fraction of votes that are given to the most voted solution $Sol(\mathcal{S}, \{A_i\})$. The larger fraction of votes for the predicted solution, the larger the number of agents that have voted for the predicted solution or the larger the number of cases that each individual agent has retrieved endorsing the predicted solution, and therefore the higher the confidence on having predicted the correct solution.

The *Halting* decision policy D_H is defined using the same competence model: $D_H(\mathcal{R}^c) = (C\text{-}Competence(\mathcal{R}^c) \geq \eta)$, where η is a threshold parameter. This policy determines that the convener agents stops inviting agents when $D_H(\mathcal{R}^c) = true$. The intuition behind the D_H decision policy is that if the confidence on the solution predicted by the current committee is high enough, there is no need for inviting more agents to join the committee. Notice that when A_i is alone (and can be considered as a committee of 1) this decision is equivalent to choose between solving the problem individually or convening a committee. In our experiments we have set $\eta = 0.75$.

6 Experimental evaluation

This section presents the experimental evaluation of the performance of PB-CCS. To evaluate the behavior of PB-CCS using the learnt competence models we have compared it against the CCS and the B-CCS. We have made experiments with $\mathcal{M}$AC systems composed of 3, 5, 7, 9, 11, 13, and 15 agents. Moreover, the agents use a standard 3-Nearest Neighbor (k-NN) [7]; in our experiments $k = 3$, that is to say the retrieval set of a CBR agent will consists of three cases. Since PB-CCS only requires that the CBR agents provide a retrieval, any other CBR technique that complies could have been used in the experiments, or a different value of k could be used. The only difference on using one CBR technique or parameter over another is the average prediction accuracy of individual agents; however, for the purpose of evaluating PB-CCS, any one of the compliant CBR techniques is suitable, since we will focus on the accuracy *improvement* over the accuracy provided to the individual agents by whatever CBR technique is being used in the experiments.

We have designed an experimental suite with a case base of 280 marine sponges pertaining to three different orders of the *Demospongiae* class (*Astrophorida*, *Hadromerida* and *Axinellida*). In an experimental run, training cases are randomly distributed among the agents. In the testing stage unseen problems arrive randomly to one of the agents. The goal of the agent receiving a problem is to identify the correct biological order given the description of a new sponge. Moreover, all the results presented here are the result of the average of five 10-fold cross validation runs.

Moreover, in order to investigate whether the proactive learning technique used in PB-CCS learns adequate competence models under different circumstances, we have performed experiments in three different scenarios: the *uniform* scenario, the *redundancy* scenario, and the *untruthful agents* scenario.

Uniform: In this scenario each individual agent receives a random sample of the training set without replication of cases (i.e. the case bases of the agents are disjoint).

Redundancy: In this scenario each agent receives a random sample with replication of cases (i.e. two agents may own the same case). To measure the degree of redundancy introduced, we will define the redundancy R as follows:

$$R = \frac{\left(\sum_{i=1\cdots n} \#(C_i)\right) - N}{N * (n-1)}$$

where n is the number of agents, $N = \#(\cup_{i=1\cdots n} C_i)$ is the total number of different cases in the system, and C_i is the individual case base of the agent A_i.

When the individual case bases of the agents are disjoint there is no redundancy ($R = 0$) since the numerator is zero because $N = \#(\cup_{i=1\cdots n} C_i) = \sum_{i=1\cdots n} \#(C_i)$. Moreover, when all the individual case bases of the agents are identical (all the agents own the same cases) the redundancy is maximal, and thus $R = 1$, since if all the case bases are identical, then $\forall_j \cup_{i=1\cdots n} C_i = C_j$, and thus $\#(C_j) = N$.

In this scenario we have used a degree of redundancy of $R = 0.1$. The data set that is distributed among the agents has 280 cases (as we perform a 10-fold cross validation, there training set to distribute among the agents at each fold has 254 cases). To better understand what $R = 0.1$ represents, consider this: in a 5 agents scenario with $R = 0.0$ each agent will receive, on average, 50.4 cases (since the 280 cases in the data set are divided in a training set of 252 cases and a test set of 28 cases during the 10-fold cross validation, and $252/5 = 50.4$). Moreover, with $R = 0.1$ each agent will receive, on average, 70.54 cases since some of the training cases will be replicated among the agents case bases (if $R = 1.0$ each agent will receive the 252 training cases). In a 9 agents scenario, with $R = 0.0$ each agent will receive about 28.00 cases, and with $R = 0.1$ each agent will receive 50.4 cases in average.

Untruthful Agents: In this scenario some of the agents in the committee are untruthful, i.e. when an agent asks them for help, they will sometimes answer a solution different from their true individual prediction (i.e. they will lie). Nevertheless, those agents answer the truthful solution when they are in the role of the convener agent.

The goal of performing experiments in these scenarios is to test whether the individually learnt competence models are useful to decide when to stop inviting agents to join the committee and which agents to invite under different conditions. The uniform scenario is the basic scenario, where each individual agent has a different sample of the training set. In the redundancy scenario, since each agent has more cases than in the uniform scenario, it is expected that each individual agent will achieve a greater individual accuracy. Therefore, we expect that the number of times PB-CCS solves a problem individually without need to convene a committee increases in the redundancy scenario. Moreover, the average number of agents needed to solve a problem should decrease for the same reason. However, only if the proactive learning captures the properties of this scenario will these tendencies be apparent on the experimental results.

Finally, the untruthful agents scenario models a situation in which not all the agents of the system can be trusted. We have designed this scenario to test whether the learnt competence models can detect which agents in the system can be trusted and which cannot. In this scenario, we expect that the performance of CCS decreases with respect to the uniform scenario (since individual prediction accuracy on average will diminish). However, using competence models, PB-CCS should be able to detect untruthful agents and very seldom invite them to join the committee. Consequently we expect that PB-CCS performance will not decrease as much as CCS, proving that untruthful agents have been adequately detected by the learnt competence models.

These three scenarios are evaluated on a single data set. Using several data sets would not add any more meaningful information; the only real difference between several data sets is the degree in which the ensemble effect increases the committee accuracy. However, this is not a primary concern here, since our goal is evaluating the performance of the dynamic committees with respect to convening always the full committee in a given data set. An evaluation of the ensemble effect of fixed agent committees over different data sets can be found in [25].

6.1 PB-CCS evaluation in the uniform scenario

Figure 8 shows the results for the uniform scenario. Specifically, Fig. 8a shows the committee accuracy and Figure 8b shows the average committee size. $\mathcal{M}$AC systems with 3, 5, 7, 9, 11, 13 and 15 agents are tested. For each $\mathcal{M}$AC system results for agents using CCS, B-CCS, and PB-CCS are presented. Moreover, two different parameter settings have been evaluated for PB-CCS: the first one with $\eta_1 = 0.9$ and $\eta_2 = .5$ and the second one with $\eta_1 = 0.95$ and $\eta_2 = 0.5$. In the first parameter settings the convener agent will request a confidence of at least .9 in order to stop inviting agents to join the committee, and in the second parameter settings, the convener agent will request a confidence of at least .95. Therefore, the expected behavior is that in the second parameter settings both the convened committees size and the accuracy will be larger. Moreover, both parameter settings request that all invited agents have at least a confidence of .5 of predicting the correct solution for the current problem.

Before analyzing the results shown in Fig. 8, notice that as the number of agents increases, each agent receives a smaller case base. Thus, the accuracy of each individual agent is lower in the experiments with many agents. The effect of this is that the accuracy of all the collaboration strategies diminishes as the number of agents increases. However, it is important to note that this is not due to the number of agents, but to the way in which experiments have been performed, since in our experiments a larger number of agents implies smaller case bases (since the training set is divided among all the agents in the system and therefore, the more agents, the less cases that each agent receives). Therefore, MACs with large number of agents are used to model situations where data is very distributed and the agents have more incentive to cooperate.

Figure 8 shows that the accuracy of PB-CCS is very close to that of CCS. In fact, with $\eta_1 = 0.95$ the difference in accuracy between PB-CCS and CCS is not statistically significant. Moreover, the accuracy of PB-CCS (both with $\eta_1 = 0.9$ and $\eta_1 = 0.95$) is

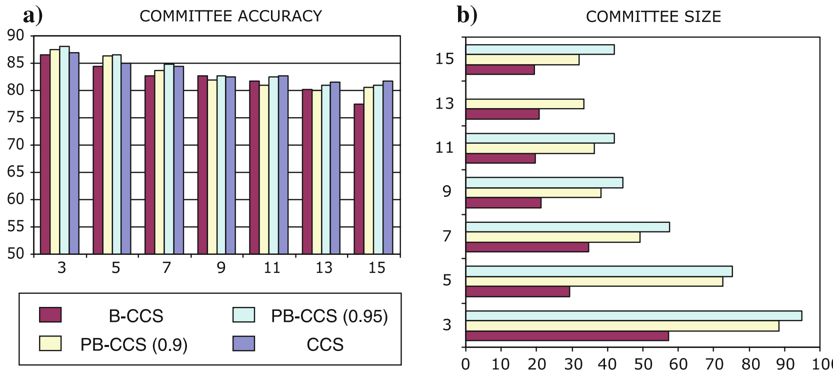


Fig. 8 Committee accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the uniform scenario

higher than the accuracy of B-CCS in all of the $\mathcal{M}$AC systems except in the 9 agents system (where the difference is not statistically significant).

Figure 8b shows the average size of the committees convened by PB-CCS and B-CCS expressed as the percentage of the agents in the $\mathcal{M}$AC system convened in average (we do not show the size of the committees convened by CCS that is always 100%). The figure shows that the average size of the committees convened by PB-CCS is smaller than the committees convened by CCS, specially in $\mathcal{M}$AC systems with a large number of agents. The figure also shows that the average size of the committees convened by PB-CCS is larger than in B-CCS. This proves that PB-CCS invites more agents to join the committee precisely when they are needed to improve performance (since PB-CCS has a higher accuracy than B-CCS). Moreover, the threshold parameter η_1 affects the average size of the committee: if $\eta_1 = 0.95$ the size of the committees tends to be larger than with $\eta_1 = 0.9$, as expected.

PB-CCS achieves a good tradeoff of accuracy and committee size since the accuracy achieved by PB-CCS with $\eta_1 = 0.95$ is undistinguishable of the accuracy of CCS while the average size of a committee convened by PB-CCS is much smaller than 100% (the size of a committee convened by CCS). B-CCS also achieves a good tradeoff since the accuracy values are only a bit lower than that of CCS, and the committee size is much smaller than that of CCS. Notice that the only difference between PB-CCS and B-CCS is that in PB-CCS agents learn their own competence models, and in B-CCS competence models have to be predefined. The competence models used by B-CCS in these experiments have been hand-tuned for the uniform scenario, and thus B-CCS performs quite well.

Figure 9 shows the percentage of times that the convener agent has convened committees of different sizes with $\eta_1 = 0.9$. An horizontal bar is shown for each $\mathcal{M}$AC system. Each bar is divided in several intervals: the leftmost interval represents the percentage of times that the convener agent has solved the problem individually; the second interval represents the percentage of times that a committee of 2 agents has been convened, and so on. The right most interval represents the percentage of times that a committee containing all the agents in the system has been convened. Figure 9

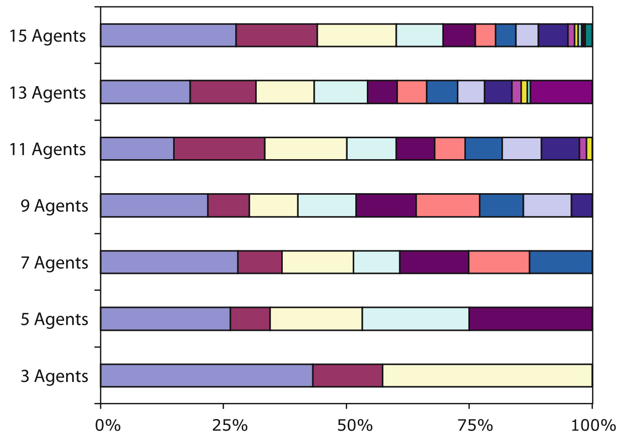


Fig. 9 Percentage of times that the convener agent has convened committees of different sizes in the uniform scenario using PB-CCS with $\eta_1 = 0.9$

shows that in the 3 agents system, about 40% of the times the convener agent solves the problem individually without the need of convening a committee. Clearly, this percentage is reduced in the $\mathcal{M}$AC systems with more agents because individual case bases are smaller and the individual accuracy is lower. Concerning the average size of the committees convened, PB-CCS is able to convene smaller committees than CCS without compromising accuracy. For instance, in the 11 agents $\mathcal{M}$AC 50% of times convenes committees of 3 or less agents. Thus, these results show that the proactive learning is able to learn adequate competence models.

Summarizing, we have shown that using PB-CCS a reasonable number of times problems can be solved individually by an agent without recourse to a committee while maintaining the average accuracy. Moreover, when committees are needed, they are convened, but with a smaller size than the 100% committee of CCS. Consequently the proactive learning process is acquiring adequate competence models (since they exhibit the expected behavior). Moreover, we have seen that varying parameters η_1 and η_2 have the expected result in the behavior of PB-CCS since $\eta_1 = 0.95$ achieves a higher accuracy (and slightly larger committees) than $\eta_1 = 0.9$.

6.2 PB-CCS evaluation in the redundancy scenario

In the redundancy scenario the case bases of the individual agents are not disjoint as in the uniform scenario, but have some overlapping, i.e. there are cases that are present in more than one agents' case base. Moreover, we have used $\eta_1 = 0.9$ and $\eta_2 = 0.5$ for all the experiments in the redundancy scenario.

Figure 10 shows the results for the redundancy scenario. Figure 10a shows that the accuracy of PB-CCS, B-CCS, and CCS are very similar, and their accuracy values are higher than those achieved in the uniform scenario, as expected for the presence of redundancy. In fact, the difference in accuracy is only statistically significant in the 11, 13, and 15 agents systems where B-CCS achieves a lower accuracy than PB-CCS and CCS. Therefore, PB-CCS is as proficient as CCS.

In terms of committee size, PB-CCS convenes much smaller committees than the 100% committee of CCS as Fig. 10b shows. Again, this is specially noticeable in $\mathcal{M}$AC systems with a large number of agents. For instance, in a $\mathcal{M}$AC system with 13 agents, less than the 30% of the agents are convened in average, while CCS always convenes

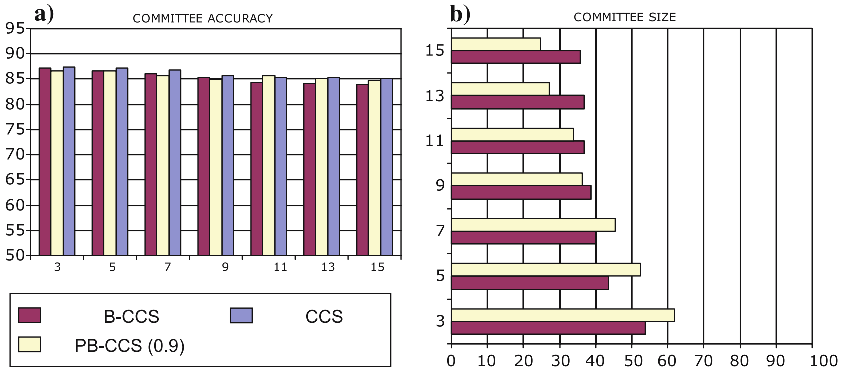


Fig. 10 Committee accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the redundancy scenario

the 100% of the agents. Comparing the behavior of B-CCS and PB-CCS in the redundancy scenario with their behavior in the uniform scenario, it would be expected that they convene smaller committees in the redundancy scenario since individual agents have higher accuracy. PB-CCS shows exactly this behavior, i.e. it convenes smaller committees in the redundancy scenario while maintaining the accuracy. However, B-CCS convenes larger committees in the redundancy scenario than in the uniform scenario. This happens because the competence models used by B-CCS are predefined, and do not change from one scenario to the other. This shows that learning competence models, as PB-CCS does, is a clear advantage.

Concerning the behavior of B-CCS, Fig. 10 shows an interesting fact. As we previously said, B-CCS uses predefined competence models that in these experiments where hand-tuned to perform well in the uniform scenario. Figure 10 clearly shows that the behavior of B-CCS degrades as the number of agents increase (i.e. B-CCS achieves lower accuracy compared with PB-CCS or CCS and convenes larger committees than PB-CCS as the number of agents increase). This effect can be explained by the fact that in the redundancy scenario experiments we have fixed a redundancy of $R = 0.1$; however $R = 0.1$ does not represent the same amount of redundancy in the 3 agents system than in the 15 agents system. In fact, a redundancy of $R = 0.1$ in a 15 agents system is a huge degree of redundancy. Therefore, the larger the number of agents in the redundancy scenario, the further away we are from the uniform scenario, and thus the further we are from the scenario for which the competence model of B-CCS was designed (and thus, the worse B-CCS performs). Therefore, we can see that learning is better because acquires competence models adapted to the current scenario, while predetermined competence models would require hand-tuning for *each* scenario.

Finally, Fig. 11 shows the percentage of times that the convener agent has convened committees of different sizes in the redundancy scenario. Fig. 11 shows that in the redundancy scenario, agents using PB-CCS solve problems individually more often than in the uniform scenario (shown in Fig. 9). Therefore the proactive learning process has acquired good competence models, since the behavior of PB-CCS is the expected one, i.e. convenes smaller committees in the redundancy scenario since

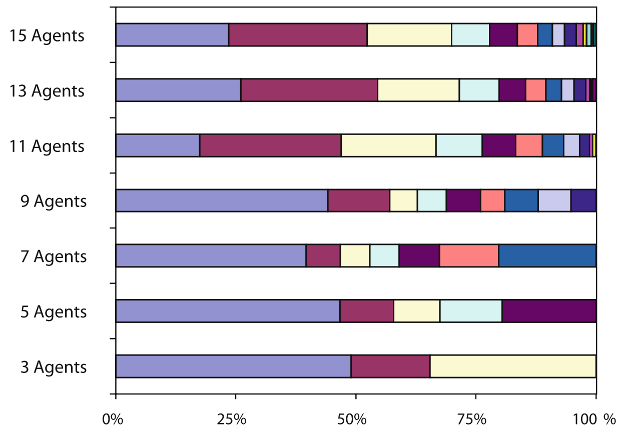


Fig. 11 Percentage of times that the convener agent has convened committees of different sizes in the redundancy scenario using PB-CCS with $\eta_1 = 0.9$

since if the individual accuracy is higher, the agents will individually solve problems correctly more often, and therefore, a committee has to be convened less often. For instance, in $\mathcal{M}$AC systems composed of 9 agents or less, agents solve problems individually between a 40% and a 50% of the times. Moreover, if there is need to convene a committee, PB-CCS convenes smaller size committees than in the uniform scenario; for instance, in the 11 agents $\mathcal{M}$AC, the 46.66% of the times the committee size is one or two agents.

Summarizing, we have seen that proactive learning of competence models gives PB-CCS an adaptive behavior (that B-CCS lacks, since uses predefined competence models). Moreover, we have also seen that PB-CCS has effectively detected (thanks to proactive learning) that in the redundancy scenario an agent can solve problems individually without recourse to a committee more often than in the uniform scenario. Similarly, PB-CCS is able to convene smaller committees because the acquired competence models capture the fact that individual predictions are more accurate.

6.3 PB-CCS evaluation in the untruthful agents scenario

The untruthful agents scenario has two goals: the first one is to evaluate the robustness of PB-CCS in the presence of malicious agents (that is equivalent to evaluate the robustness of PB-CCS to noise); the second goal is to evaluate whether the proactive learning process produces adequate competence models, i.e. competence models that can detect that some agents have a very low confidence (the untruthful agents).

Specifically, these *untruthful* agents will lie about their predictions 50% of the times to other conveners, but will not lie to themselves when they are themselves convener agents. Specifically, there will be 1–7 untruthful agents in the 3, 5, 7, 9, 11, 13 and 15 agents systems respectively. Moreover, in this scenario we expect that the Proactive Bounded Counsel Agent Selection decision policy, D_{AS}, is able to effectively decide which agents have a high confidence and which ones have a low confidence, so that untruthful agents are very seldom invited to join a committee. Finally, the accuracy of all the collaboration strategies is expected to be lower than in the uniform or redundancy scenarios since there are less agents with high confidence in the system that can be invited to join the committee.

Figure 12 shows the results for the untruthful agents scenario, with threshold parameters $\eta_1 = 0.9$ and $\eta_2 = .5$. Fig. 12a shows that in this scenario the accuracy achieved by CCS and B-CCS is lower than the accuracy achieved by PB-CCS. The accuracy of CCS is lower than that of B-CCS since CCS always invites the untruthful agents to join the committee, while B-CCS does not. Moreover, the accuracy of the three collaboration strategies in this untruthful agents scenario is lower than that of the uniform scenario (Fig. 8). This reduction in accuracy is expected, since the presence of untruthful agents leaves less truthful agents to form committees with , and thus the maximum accuracy that can be reached by virtue of the ensemble effect is lower.

Since CCS does not perform any agent selection, all the untruthful agents are convened and CCS accuracy drops from 81.71% to 66.80% in the 15 agents scenario. Thus, we can conclude that CCS is not robust when there are agents that cannot be trusted. B-CCS is also not robust, because it does not select agents, only decides when to halt inviting agents to the committee. Consequently, depending on the members of an existing committee, on some occasions they will all be truthful (and B-CCS will stop as it would do in the uniform scenario) while on other occasions the committee

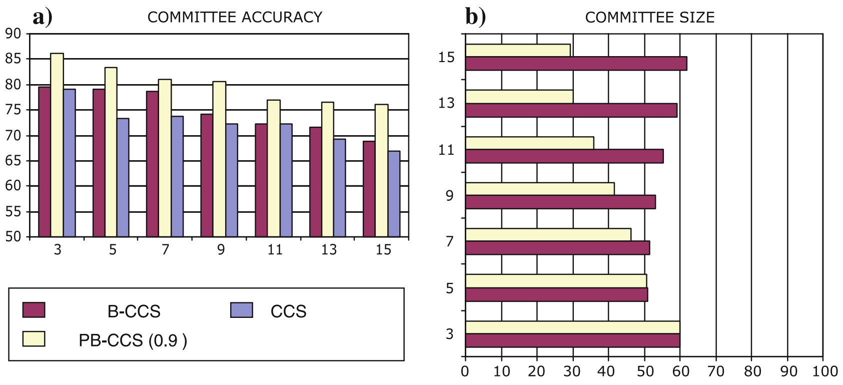


Fig. 12 Committee accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the untruthful agents scenario

will have one or more untruthful agents that will decrease the committee confidence, causing B-CCS to invite more agents than in the uniform scenario.

PB-CCS uses an agent selection policy, and as Fig. 12a shows, the accuracy of PB-CCS is much higher than that of B-CCS and CCS. Therefore PB-CCS is much more robust since the untruthful agents are much less often invited to join the committee. However, the accuracy of PB-CCS decreases with respect to the uniform scenario because there are less agents to convene committees with (since untruthful are avoided), and not because of a bad agent selection policy, as we will later show.

Figure 12b shows the average committee sizes, and shows that PB-CCS convenes smaller committees than B-CCS. Moreover, as the number of agents increases, the difference in size of the committees convened by B-CCS and PB-CCS increases. The explanation is that PB-CCS learns adequate competence models for the D_{AS} decision policy, since it selects truthful agents to join the committee and very seldom invites untruthful agents. For instance, in the 13 agents $\mathcal{M}$AC, PB-CCS all convened committees have 10 agents or less, and 75% of times only 4 or less agents are convened.

Figure 13 shows the percentage of times that committees of different sizes have been convened in the untruthful agents scenario. Specifically, PB-CCS tends to convene smaller committees than in the uniform scenario (Fig, 9); the reason is that there are less agents with a high confidence that can be invited to join the committee. Notice that often agents are able to individually solve problems in this scenario (in fact, with the same frequency as in the uniform scenario). Therefore, the capability of an agent to determine when it can solve problems individually is clearly beneficial, since it lowers the negative effects due to the presence of untruthful agents.

For the purpose of assessing the degree in which the Proactive Bounded Agent Selection decision policy D_{AS} is able to detect the untruthful agents, the number of times that each agent has been invited to join a committee has been counted, summarized in Table 2. For each $\mathcal{M}$AC system, two values are shown: the average number of times that a truthful agent has been convened to a committee and the average number of times that an untruthful agent has been convened to a committee. For instance, in the 3 agents $\mathcal{M}$AC system, each one of the two truthful agents is invited to join a committee a 47.57% of the times while the only untruthful agent is only invited to join a committee 5.07% of the times. This clearly shows that D_{AS} selects a

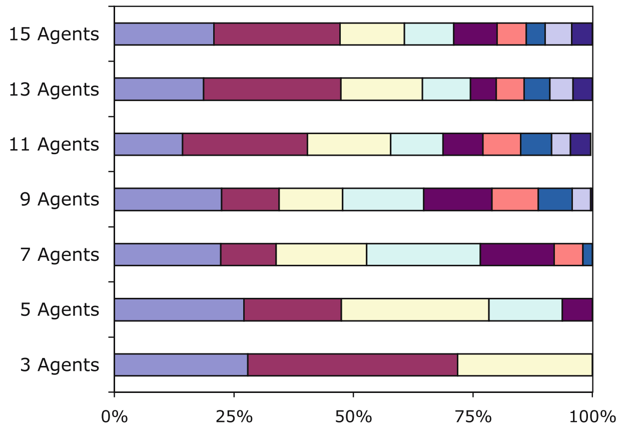


Fig. 13 Percentage of times that the convener agent has convened committees of different sizes in the untruthful scenario using PB-CCS with $\eta_1 = 0.9$

truthful agent much more often. In fact, the degree to which D_{AS} is able to detect the untruthful agents depends of the threshold parameter η_2. In these experiments we have set $\eta_2 = .5$, but if we set a higher value (e.g. $\eta_2 = .75$) untruthful agents would be invited even less often. Notice that $\eta_2 \geq .5$ in order to preserve one the preconditions of the ensemble effect, namely that the individual error of the individual classifiers must be lower than .5.

Summarizing, this scenario also shows that the proactive learning of competence models gives PB-CCS an adaptive behavior. As we said before, this scenario was designed to validate whether the learnt competence models would able to detect untruthful agents. The experimental results show (a) that the decision to solve a problem individually without recourse to a committee is not affected by untruthful agents, and (b) that the untruthful agents are effectively detected because they are very seldom invited to join committees. Notice that an untruthful agent is only rejected if their estimated confidence is lower than $\eta_2 = .5$. However, competence models are learnt from a finite set of examples, and this involves some degree of error: the estimated value might be sometimes slightly above .5 and some other times slightly below. The effect of this approximate value is that in the first situation untruthful agents will be invited to join the committee. Thus, the number of times an untruthful agent joins a committee (shown in the third row of Table 2) is a measure of that error. This error increases with the number of agents in the $\mathcal{M}$AC; this is as expected, since the higher the number of agents the smaller size of individual cases bases, which implies a smaller set of cases B_i used to acquire *M-examples* (Sect. 4.1).

Table 2 Average number of times that truthful and untruthful agents are invited to join a committee

Agents	3	5	7	9	11	13	15
Truthful	47.57%	44.14%	43.63%	31.0%	32.0%	32.75%	31.8%
Untruthful	5.07%	9.03%	6.82%	7.14%	9.14%	11.57%	11.08%

6.4 Proactive bounded counsel cost

Proactive learning of competence models has a computational cost for the agents. Specifically, the agents send a set of problems to the other agents (in order to evaluate their predictions), and also solve individually another set of problems. However, notice that if proactive learning is not used, the cost of acquiring the competence models does not disappear, but it is merely shifted to a previous phase where a good competence model (or any other policy to decide how to form committees) is built by hand (as exemplified in Sect. 5). Moreover, since a predetermined model is not valid for all scenarios, either a static scenario has to be assumed or a range of predetermined models would be needed. The adaptive nature of proactively learning competence models is thus reducing cost in practice.

Moreover, in a real system, the cost of acquiring competence models can be greatly reduced by interleaving proactive learning with the regular process of solving problems. Imagine that a specific agent wants to learn a competence model; the agent can solve the problems that arrive from users convening committees using any other collaboration strategy (such as CCS), and store the predictions that the other agents during the regular solution of problems. Once the agent has collected those problems, it can construct the *M-examples* in the same way as explained in Sect. 4.1 and then learn the corresponding competence models. In other words, it is not necessary for an agent to collect all the required *M-example* in one step as we have made in these experiments for the sake of presentation clarity, but they can be automatically acquired during the regular process of solving problems.

6.5 Conclusions from the experiments

Finally, we would like to provide a summary of the conclusions we can draw from all the experiments presented in this section (including all the three scenarios). First of all, we have seen that agents can individually solve problems often without compromising accuracy, and PB-CCS is a strategy that allows them to do that. Second, depending on the scenario, the conditions under which an agent can solve problems individually may vary; the advantage of learning is that it is adaptive to these changes. Indeed, in the redundancy scenario (where individual accuracy is higher) PB-CCS accurately determines that a problem can be solved individually without recourse to a committee more often. Third, also depending on the scenario, the size of a committee that achieves high accuracy varies; we have seen that PB-CCS is able to detect that and convene adequate committees for each scenario. Finally, we have also seen that deciding which agents to collaborate with is an important issue, and proactive leaning has proven to be adaptive enough to adequately select which agents to collaborate with. As a global conclusion, the experiments show that proactive learning allows the agents to learn when and with which agents to collaborate, and increases the degree of autonomy of agents in the sense they do not require a human to design specific competence models for them.

7 Related work

The main areas related to our work are ensemble learning, distributed CBR, and agent team formation.

Concerning ensemble learning, the “ensemble effect” is a general result on multiple model learning [15] (already introduced in Sect. 1.1). The BEM (*Basic Ensemble Method*) is presented in [26] as a basic way to combine continuous estimators, and since then many other methods have been proposed: *Stacking generalization* [30], *Cascade generalization* [13], *Bagging* [4] or *Boosting* [12] are some examples. However, ensemble methods assume a centralized control of all the data while this is not true in our approach. Ensemble methods assume that all data is available to a centralized algorithm that constructs the individual predictors that form the ensemble. In our approach, each agent is the owner of its individual data (each individual agent has only access to the data contained in its own case base, and has no access to the data in other agents’ case bases), and the distribution of data among agents cannot be modified by a centralized algorithm since this will violate the autonomy of the agents. The control in $\mathcal{M}$AC systems is decentralized, that is to say, the global effect is achieved by the individual decisions taken by each agent, while in ensemble learning all the decisions are made by a central agency (the ensemble learning method).

Meta-learning [6] is the more similar of the ensemble learning methods to our approach. The meta-learning approach assumes a scenario where control is centralized, the base predictors have access to disjoint data sets, and the meta learner has access to all data. The goal is to learn a meta-classifier whose training data is the outcomes of the base predictors; in order to do so, they require access to the whole data set. Thus, this approach is not directly applicable to our scenario because this assumption does not hold in a system where agents are autonomous. The final result is an *arbitrator tree*, a centralized method whose goal is to improve accuracy; in our approach, each agent learns a collection of confidence trees as competence models used to take a variety of decisions.

In general, the main difference between ensemble learning and our approach concerns the goals and assumptions of each approach. The goal of ensemble learning is to maximize accuracy by complex algorithms that have the freedom to manipulate all data using intelligent and informed techniques; for this purpose they assume a centralized control and access over all data. Our goal is to develop techniques capable of exploiting the ensemble effect assuming that control is decentralized and data is distributed.

Another related area is that of distributed CBR systems. McGinty and Smyth [18] present collaborative case-based reasoning (CCBR) as a framework where experience is distributed among multiple CBR agents. Their individual agents are only capable of solving problems that fall within their area of expertise. When an agent cannot solve a problem, it broadcasts the problem to the rest of agents, and if there is some agent capable of that problem, that agent will return the retrieved cases to the initial agent. This approach differs from ours in that they only perform case retrieval in a distributed way. The initiating agent receives, for a particular problem, the cases in the retrieval sets of the rest of the agents, and then it solves the problem individually using those newly acquired cases. In our approach, an agent can only work with its individual case base since no agent has access to the cases owned by another agent. Thus, while the CCBR approach can be seen as a distributed-retrieval approach, our approach can be seen as a distributed-reuse approach (i.e. in our approach, collaboration takes place when reusing the retrieved cases to decide a joint prediction).

Another related approach is multi-case-base reasoning (MCBR) [16, 17], deals with distributed systems where there are several case bases available for the same task.

Moreover, each case base may not correspond to exactly the same task domain, or may reflect some different user preferences, etc. Therefore cases must be adapted to be transferred from one case base to another. Moreover, the main difference between our approach and MCBR is again that they focus on distributed retrieval.

Concerning agent team formation, the main difference between the team formation literature and our work is the difference between teams and committees. On the one hand, in committees we assume that all participant agents have different backgrounds but they are all capable of achieving the overall goals of the task at hand; thus committees are formed to improve the quality of the joint solution to be adopted. On the other hand, in teams we assume that individual agents have different capabilities, but none of them individually is able to achieve the overall goals. For instance, the ORCAS framework [14] uses CBR to configure teams of agents in such a way that they achieve the overall goals for a particular problem.

Another approach to team formation is the Cooperative Problem Solving (CPS) framework [8, 31]. Four stages are clearly identified in CPS: (1) potential recognition (finding which agents can perform certain tasks), (2) team formation, (3) plan formation, and (4) plan execution. This framework is certainly a general way to deal with team formation, however it focuses on finding a plan (or protocol) that the individual agents can follow to collaborative solve a given problem by combining their capabilities, and usually assumes that if two different agents are able to perform a task, it does not matter which of both is selected. In our framework, we deal with a more specific form of collaboration (committees), where all the agents are capable of predicting solutions, but the selection of the specific members of the committee is crucial for improving the performance of the committee.

Also relevant is work on learning to form coalitions of agents by Sarathi and Sen [9], where they propose a framework for agents that learn who are the best agents to collaborate with in the form of stable coalitions. However, they focus on the assignment of tasks to individual agents that can perform them in a more efficient way, rather than aggregating individual predictions as we do.

8 Conclusions and future work

We have presented a framework for collaborative multi-agent CBR systems called $\mathcal{M}$AC. The framework is collaborative in the sense that the agents collaborate with other agents if this can report some improvement in performance. This article addresses two main issues on collaboration: when to collaborate, and with whom to collaborate. We have presented the idea of committees to study these issues, and specifically we presented a collaboration strategy called PB-CCS that allows the agents to learn when to convene committees, and which agents to invite to each committee. We have also presented a proactive learning technique that allows an agent to learn its individual competence models, that are required by PB-CCS.

The empirical evaluation we performed supports the proactive learning approach we have taken: first, learning competence models (PB-CCS) is more robust than predetermined component models (B-CCS) and fixed committees (CCS), since PB-CCS achieves higher accuracy values in a wider range of scenarios than CCS or B-CCS. The robustness of PB-CCS is achieved by the capability to adapt to the properties of different scenarios provided by the proactive learning of competence models. Predefined models are not adaptive to the different scenarios, and require a process of hand-tuning for each foreseeable scenario. Learning, however, increases agent autonomy in that they are capable of acquiring the specific competence models adequate for the scenario they have to deal with.

Second, PB-CCS increases the agents autonomy by providing the ability to decide when they can solve problems individually and when they are better off convening a committee. Moreover, the proactive learning process allows the agents to take these decisions adequately in a range of scenarios. The frequency in which an agent can solve problems individually depends on its individual accuracy; for instance, in the redundancy scenario, where individual accuracy is higher, agents are capable to adapt to this circumstance and individually solve problems more often.

Third, PB-CCS learns to convene committees with a size that is not larger than what is needed to maintain the committee accuracy. For instance, in the redundancy scenario, where individual accuracy is higher, PB-CCS is capable of convening, on average, smaller size committees. Moreover, in the untruthful scenario PB-CCS convenes smaller size committees because it learns that certain agents are unreliable. And fourth, the proactive learning process acquires adequate competence models since PB-CCS behaves as expected in all the three scenarios.

Moreover, given the experimental results, we can say that PB-CCS will perform well (i.e. having a high accuracy) if (a) the agents have a reasonable number of cases (needed to collect M-examples), (b) the agents do not change their behavior radically (otherwise the competence models wouldn't predict well their behavior), and (c) there are at least some competent and truthful agents in the system (otherwise no collaboration strategy can perform well).

As future work, we plan to perform incremental learning, where the competence models should be updated as time passes. In this scenario, the competence models should be able to adapt if more agents enter or leave in the $\mathcal{M}$AC system, and to reflect changes in the types of problems that the system is solving. If a convener agents store the SERs received from the other agents, the competence models could be updated by learning new trees reflecting the changes in the behaviors of the other agents. To detect when a competence model has to be updated, an agent could compare the behavior of another agent with the predicted behavior from the learned competence model for that agent. When the learned competence model does not predict well the behavior of that agent anymore, it has to be updated.

We also intend to enrich our view of committees as an electronic institution. As we mentioned in the introduction, committees have the discussion and deliberation phases, and we have now focused on the second one, where voting among alternatives is performed. The first phase involves the presentation, justification and argumentation of those alternatives. We think this phase can be modelled as an argumentation process among agents (amenable to be specified using the methodology of electronic institutions and ISLANDER) where the alternatives are justified, criticized and modified. As a first step in this direction, we have proposed a framework where the alternatives can be criticized by counterarguments and counterexamples and the agents are able to modify the alternatives they present using the information argued by other agent [24].

Acknowledgements The authors thank Josep-Lluís Arcos of the IIIA-CSIC for their support and for the development of the Noos agent platform, and Eva Armengol for the marine sponge data set, created and maintained by her. Support for this work came from project CBR-ProMusic TIC2003-07776-C02-02.

References

1. Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications, 7*(1), 39–59. online at <url:http://www.iiia.csic.es/People/enric/AICom_ToC.html>.
2. Aha, D. (Ed.) (1997) *Lazy learning*. Kluwer Academic Publishers.
3. Brams, S. J., & Fishburn, P. C. (1983). *Approval voting*. Boston: Birkhauser.
4. Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.
5. Cestnik, B., & Bratko, I. (1991). On estimating probabilities in tree pruning. In *Machine learning-European working session on learning-91*, Vol. 482 of *Lecture Notes in Artificial Intelligence*. (pp. 151–163). Springer Verlag.
6. Chan, P. K., & Stolfo, S. J. (1995). A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the 12th international conference on Machine learning*, pp. 90–98.
7. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification'. *IEEE Transactions on Information Theory 13*(1), 21–27.
8. Dignum, F., Dunin-Kęplicz, B., & Verbrugge, R. (2001). Agent theory for eam formation by dialogue. *Lecture Notes in Computer Science, 1986*, 150–166.
9. Dutta, P. S., & Sen, S. (2002). Emergence of stable coalitions via task exchanges. In C. Castelfranchi, & W. L. Johnson (Eds.), *Proceedings of 1st international conference on automous agents and multiagent systems*, pp. 312–313.
10. Esteva, M., Padget, J., & Sierra, C., (To appear). Formalising a language for institutions and norms. In *Intelligent agents VIII, proceedings ATAL'01*.
11. Esteva, M., Rodriguez-Aguilar, J. A., Sierra, C., Garcia, P., & Arcos, J. L. (2001). On the formal specification of electronic institutions. In *Agent mediated electronic commerce*, Vol. 1991 of *LNAI*. Springer-Verlag.
12. Freund, Y., & Schapire, R. E. (1996). Experiments with a new Boosting algorithm. In *Proceedings 13th international conference on machine learning*, pp. 148–146.
13. Gama, J. (1998). Local cascade generalization. In *Proceedings 15th international conference on machine learning*, pp. 206–214.
14. Gomez, M., & Plaza, E. (2004). Extending matchmaking to maximize capability reuse. In *Proceedings of the AAMAS 2004*, pp. 144–152.
15. Hansen, L. K. & Salamon, P. (1990). Neural networks ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 993–1001.
16. Leake, D. B. & Sooriamurthi, R. (2001). When two case bases are better than one: Exploiting multiple case bases. In *ICCBR*, pp. 321–335.
17. Leake, D. B. & Sooriamurthi, R. (2002). Managing multiple case bases: Dimensions and issues. In *Proceedings of the 15th international Florida Artificial Intelligence Research Society (FLAIRS)*, pp. 106–110.
18. McGinty, L., & Smyth, B. (2001). Collaborative case-based reasoning: Applications in personalized route planning. In *Case based reasoning ICCBR-01*, pp. 362–376.
19. Ontañón, S., & Plaza, E. (2002a). A bartering aproach to improve multiagent learning. In *1st International joint conference in autonomous agents and multiagent systems*.
20. Ontañón, S., & Plaza, E. (2002b). Collaboration strategies to improve multiagent learning. *Lecture Notes in Artificial Intelligence, 2430*, 331–344.
21. Ontañón, S., & Plaza, E. (2003a). Collaborative case retention strategies for CBR agents. *Lecture Notes in Artificial Intelligence, 2689*, 392–406.
22. Ontañón, S., & Plaza, E. (2003b). Justification-based multiagent learning. In *International conference on machine learning ICML-2003*, pp. 576–583.
23. Ontañón, S. & Plaza, E. (2005). Recycling data for multi-agent learning. In *Proceedings 22nd international conference on machine learning ICML-2005*, pp. 633–640.
24. Ontañón, S., & Plaza, E. (2006). Arguments and counterexamples in case-based joint deliberation. In *Proceedings AAMAS'06 workshop on argumentation on multi-agent systems*, p. to appear.
25. Ontañón, S., (2005). Ensemble case based learning for multi-agent systems. Ph.D. thesis, Universitat Autònoma de Barcelona.
26. Perrone, M. P., & Cooper, L. N. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In *Artificial neural networks for speech and vision*, Chapman-Hall.
27. Plaza, E., & Ontañón, S. (2001). Ensemble case-based reasoning: Collaboration policies for multiagent cooperative CBR. In I. Watson, & Q. Yang (Eds.), *In case-based reasoning research and development: ICCBR-2001*, pp. 437–451.

28. Plaza, E., & Ontañón, S. (2003). Cooperative multiagent learning. *Lecture Notes in Artificial Intelligence, 2636*, 1–17.
29. Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.
30. Wolpert, D. H. (1990). Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM.
31. Wooldridge, M., & Jennings, N. R. (1994). Towards a theory of cooperative problem solving. In *Proceedings modelling autonomous agents in a multi-agent world (MAAMAW-94)*. (pp. 15–26). Odense, Denmark.